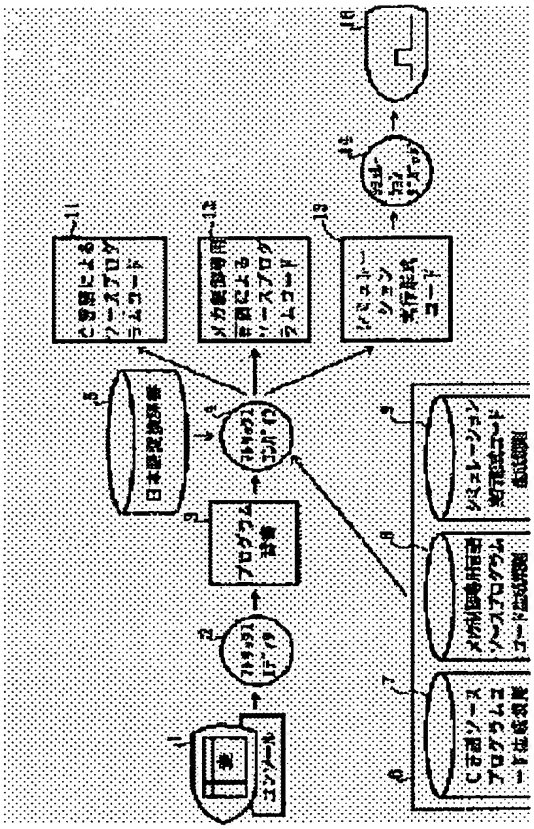
PROGRAM DEVELOPMENT SUPPORTING SYSTEM

Patent number: JP9016382
Publication date: 1997-01-17
Inventor: OTSUKI JUNICHI; IKETANI TAKESHI
Applicant: OKI ELECTRIC IND CO LTD
Classification:
- international: G06F9/06
- european:
Application number: JP19950187894 19950630
Priority number(s):

Abstract of **JP9016382**

PURPOSE: To automatically provide a source program code by easily and simply describing a device integrated software. CONSTITUTION: A program to be inputted by using a matrix editor 2 is expressed in a table format just like a tabulation software. When processing described in each column of a table visually described on a WS is successively executed from the head, the program can be executed. According to fixed rules, a matrix compiler 4 converts a program dictionary 3 corresponding to this table or the other definition drawing to the source program code in a C language, for example. Besides, the source program in a language dedicated to mechanism control is generated from the speciality of firmware as well.



資料①

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平9－16382

(43)公開日　平成９年(1997)１月17日

(51)Int.Cl.	識別記号	ＦＩ		
G06F 9/06	530	G06F 9/06	530	G

審査請求　未請求　請求項の数４　ＦＤ　（全23頁）

(21)出願番号　特願平7－187894

(22)出願日　平成７年(1995)６月30日

(71)出願人　000000295
沖電気工業株式会社
東京都港区虎ノ門１丁目７番12号

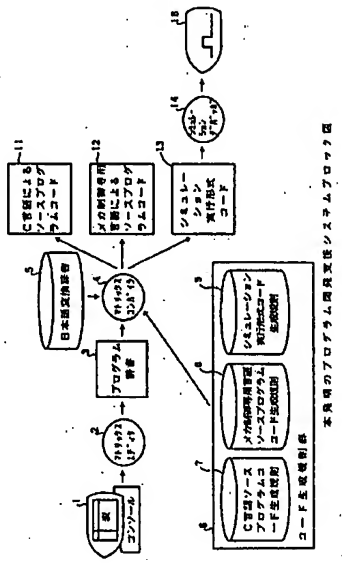
(72)発明者　大槻　純一
東京都港区虎ノ門１丁目７番12号　沖電気工業株式会社内

(72)発明者　池谷　健
東京都港区虎ノ門１丁目７番12号　沖電気工業株式会社内

(74)代理人　弁理士　佐藤　幸男　（外１名）

(54)【発明の名称】プログラム開発支援システム

(57)【要約】
【目的】　装置組み込みソフトウェアを容易に簡便に記述し、自動的にそのソースプログラムコードを得る。
【構成】　マトリクスエディタ２を用いて入力するプログラムは、丁度表計算ソフトのような表形式で表現される。ＷＳ上にビジュアルに記述される表の各欄に記入された処理を先頭から順に実行すればプログラムが実行できる。マトリクスコンパイラ４は、この表やその他の定義図等に対応するプログラム辞書３を一定の規則に従って、例えばＣ言語によるソースプログラムコードに変換する。また、ファームウェアの特殊性からメカ制御専用言語によるソースプログラムコードの生成も行う。



1

【特許請求の範囲】
【請求項１】　プログラムを構成する各処理の内容を表の各欄に記入して、先頭から順にその表の内容を実行することにより、そのプログラムを実行できるように表示した表を入力するマトリクスエディタと、
このマトリクスエディタで入力され生成されたプログラム辞書に、ソースコード生成規則を適用して、前記表に対応するソースプログラムコードを生成するマトリクスコンパイラとを備えたことを特徴とするプログラム開発支援システム。
【請求項２】　プログラムを構成する各処理の内容を表の各欄に記入して、先頭から順にその表の内容を実行することにより、そのプログラムを実行できるように表示した表を入力するマトリクスエディタと、
このマトリクスエディタで入力され生成されたプログラム辞書に、シミュレータ実行形式コード生成規則を適用して、前記表に対応するシミュレーション実行形式コードを生成するマトリクスコンパイラとを備えたことを特徴とするプログラム開発支援システム。
【請求項３】　マトリクスエディタは、
モジュール定義図とデータ形式定義図と構造体定義図とデータ意味定義図と表の入力を行い、Ｃ言語によるソースプログラムコード生成のための意味解釈用のプログラム辞書を生成することを特徴とする請求項１記載のプログラム開発支援システム。
【請求項４】　メカ動作規則とシミュレータ実行形式コードとを参照して、プログラムの各命令が実行されたときの制御対象装置の各部の状態を表す状態情報を生成する状態更新部と、
前記状態情報に従って状態図を生成してプログラムの任意のステップでその状態図を表示する請求項２記載のプログラム開発支援システム。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明は、プログラム開発、特に装置組み込みソフトウェアの開発支援に利用されるプログラム開発支援システムに関する。
【０００２】
【従来の技術】ソフトウェアの１分野に装置組み込みソフトと言われるものがある。これは、ファームウェアと呼ばれている。ファームウェアはハードウェアとアプリケーションソフトウェアの中間に位置し、メカ制御、通信制御、入出力制御等のハードウェア制御を主に担当する。このファームウェアは一般のソフトウェアと比べてリアルタイム性の強い要求があり、マルチＣＰＵ、マルチタスクといった並行処理が望ましく、ハードウェアに密着しクロス開発環境を要求するといった特徴がある。従って、一般のソフトウェアがハードウェアの独立化標準化に伴ってパッケージ化の傾向にあるのに対し、ファームウェアは処理時間やハードウェア資源の強い制約のため、装置毎に最適な構成がとられ、標準化及び生産性向上が困難な特徴を持つ。
【０００３】従って、一般の汎用性の高いソフトウェアについては、各種の簡便な開発ツールが登場し利用されているが、ファームウェアの開発支援のためには、例えばプログラムのフローチャートを解析してソースプログラムのコードを得るといったものが紹介されているに過ぎず（特公平６－４０３０２号公報）、より具体的な簡便なものは実用化されていない。
【０００４】
【発明が解決しようとする課題】ところで、上記のような従来のプログラム開発支援システムには次のような解決すべき課題があった。プログラム開発支援は、プログラマーに対し比較的簡単なプログラム内容の記述を要求して、これを自動的に変換処理してソースプログラムコード等を得ることを目的とする。従って、プログラムをどう表現して開発支援システムに入力するかという点が問題となる。従来、このようなプログラムの表現が専門的な詳細な知識を必要とし、多くの変数の値に対する条件を満たすバグの無いプログラムを作成するためには、高度に熟練をする必要があった。
【０００５】また、ファームウェア開発の生産性を上げるためには、できるだけ開発済みのファームウェアを流用することが重要である。しかしながら、作成済みのソースコードプログラムは必ずしもその処理内容を正確に短時間に把握することが容易でないという問題があった。
【０００６】
【課題を解決するための手段】本発明は以上の点を解決するため次の構成を採用する。本発明のプログラム開発支援システムは、プログラムを構成する各処理の内容を表の各欄に記入して、先頭から順にその表の内容を実行することにより、そのプログラムを実行できるように表示した表を入力するマトリクスエディタと、このマトリクスエディタで入力され生成されたプログラム辞書に、ソースコード生成規則を適用して、表に対応するソースプログラムコードを生成するマトリクスコンパイラとを備える。
【０００７】本発明の別のプログラム開発支援システムは、プログラムを構成する各処理の内容を表の各欄に記入して、先頭から順にその表の内容を実行することにより、そのプログラムを実行できるように表示した表を入力するマトリクスエディタと、このマトリクスエディタで入力され生成されたプログラム辞書に、シミュレータ実行形式コード生成規則を適用して、表に対応するシミュレーション実行形式コードを生成するマトリクスコンパイラとを備える。
【０００８】なお、マトリクスエディタは、モジュール定義図とデータ形式定義図と構造体定義図とデータ意味定義図と表の入力を行い、Ｃ言語によるソースプログラ

ムコード生成のための意味解釈用のプログラム辞書を生成することが好ましい。また、メカ動作規則とシミュレータ実行形式コードとを参照して、プログラムの各命令が実行されたときの制御対象装置の各部の状態を表す状態情報を生成する状態更新部と、状態情報に従って状態図を生成してプログラムの任意のステップでその状態図を表示することが好ましい。

【0009】

【作用】マトリクスエディタを用いて入力するプログラムは、丁度表計算ソフトのような表形式で表現される。WS（ワークステーション）上にビジュアルに記述される表の各欄に記入された処理を先頭から順に実行すればプログラムが実行できる。マトリクスコンパイラは、この表やその他の定義図等に対応するプログラム辞書を一定の規則に従って、例えばC言語によるソースプログラムコードに変換する。また、ファームウェアの特殊性からメカ制御専用言語によるソースプログラムコードの生成も行う。更に、プログラムのデバッグ作業を容易にするために、シミュレーション実行形式コードを生成する。例えば、1ステップずつシミュレーションが実行されると、プログラムの制御の対象となる装置各部の状態をアニメーション化して表示する。これらにより、分かりやすく漏れが無くバグの少ないプログラム開発が可能となる。

【0010】

【実施例】以下、本発明を図の実施例を用いて詳細に説明する。

〈システム構成〉図1は、本発明のプログラム開発支援システムの具体例を示すブロック図である。図のシステムは、コンソール1と、これを用いてプログラムを表現するための表を入力するマトリクスエディタ2と、これに生成されたプログラム辞書3、マトリクスコンパイラ4、日本語変換辞書5、コード生成規則群6、マトリクスコンパイラにより生成されたC言語によるソースプログラムコード11、メカ制御専用言語によるソースプログラムコード12、シミュレーション実行形式コード13、デバッグ作業のためのシミュレーションデバッガ14等から構成される。なお、図に示すディスプレイ16はコンソール1に設けられたものを説明の都合上、右側にも示したものである。

【0011】なお、このシステムは、例えば図2に示したようなハードウェアにより実現する。図2は、本発明の実施のためのハードウェア説明図である。図の左側には、よく知られたワークステーション等のディスプレイ21、制御部本体22及びキーボード23が斜視図で表示されている。なお、キーボード23の他に、この装置を操作するためにマウス26も利用される。また、この装置の制御部本体22の内部には、図に示すように、中央処理装置（CPU）24やハードディスク装置（HDD）25等が設けられている。

【0012】図1に示すコンソール1は、図2に示したワークステーション本体を示す。また、マトリクスエディタ2やマトリクスコンパイラ4、シミュレーションデバッガ14等は、ハードディスク装置25の中に格納された処理用プログラムによって構成される。プログラム辞書3、日本語変換辞書5、コード生成規則群6や、マトリクスコンパイラ4により生成されたソースプログラムコード11，12、シミュレーション実行形式コード13等は、ハードディスク装置25やその他の記憶装置に格納される。

【0013】このシステムは、開発支援の対象となるプログラムを表示する定義図や表のような各種チャートの編集機能と、プログラムソースコードの生成機能及びシミュレーションデバッグ機能を持つ。各種チャートの編集は、マトリクスエディタ2により行う。プログラムソースコードの生成は、マトリクスコンパイラ4により行う。また、シミュレーションデバッグはシミュレーションデバッガ14により行う。

【0014】〈マトリクスエディタの機能〉本発明では、ファームウェアを如何に分かりやすく見せるかをファームウェア生産性向上のキーファクタとして捉え、ファームウェアにマッチした表現形式として表を使用した。表はテキストスタイルの表現でありながら、絵的で分かりやすい自然な表現である。表によりきちんと書いたプログラムは分かりやすく漏れがなく、バグが少ない。マトリクスエディタ2は、この目的のために、プログラマーの操作によるモジュール定義図、データ形式定義図、構造体型ページ図、データ意味定義図、表等の作成図の入力を受け付け、対応するプログラム辞書を生成する。このシステムでは、プログラミングを、プログラム解釈のための辞書を作る過程と定義する。このプログラム辞書とは、プログラマーがマトリクスエディタ2によって作るモジュール定義、データ定義や表の内容を記述した情報である。

【0015】以下、プログラムをC言語で記述するために入力すべき表の具体例を説明する。

〈表の具体例〉図3～図9は、マトリクスエディタで入力される表の具体例を示す。図3は、モジュール定義図の説明図、図4は、データ形式定義図の説明図、図5は、構造体型定義図の説明図、図6は、データ意味定義図の説明図、図7は表の例説明図である。なお、本発明においては、上記モジュール定義図、データ形式定義図、構造体型定義図、データ意味定義図及びその後で説明するマトリクス状の表を含めて単に表というように表現している。即ち、これらの表がマトリクスエディタにより入力されることによって、最終的に図1に示すマトリクスコンパイラが動作して必要なソースプログラムコード等が生成される。

【0016】図3に示すモジュール定義図は、これから開発しようとするプログラムモジュール毎の入力31と、これを処理するためのモジュール32と、出力33との概要を明らかにしたものである。即ち、例えば図に示すような入力変数A、入力変数B等を使用してモジュールで一定の演算処理を実行し、その結果を出力する。この場合の入力変数の概要やモジュールの処理概要、出力されたデータの概要を図解しておくことによって、各モジュールの役割をプログラマー本人や第三者が容易に確認できるようにする。

【0017】図4に示すデータ形式定義図では、ファイルポインタ34、変数35，36、配列37、ポインタ38等が定義され表形式に表現されている。これにより、データの形式やファイル形式を明確に表現する。この例ではファイルポインタ34に“ファイルポインタA”が設定され、変数35には整数の“変数A”が設定されるといった要領で定義が行われている。図5に示す構造体型定義図によれば、構造体型名40や、メンバ名41、メンバの型42、変数43、ポインタ44等が表形式で明らかにされている。この例では、構造体型名40には“構造体型A”が設定され、以下、同様に変数等が設定されている。

【0018】図3に示すモジュール定義図の入力31で示した入力変数は、図4に示すデータ形式定義図等によって詳細に定義される。また、モジュール32の機能は、図5に示した構造体型定義図等によって具体的に定義される。そして、その処理の結果として得られる出力33は、図6に示すようなデータ意味定義図によって明確化される。即ち、図6の出力値の概要によれば、センサの出力45の意味がその下に具体的に表現されている。即ち、出力の値が“1”であればセンサがONの状態であり、出力値が“0”の場合にはセンサがOFFの状態である。

【0019】図7に示した表は、モジュールによる具体的な処理内容を表形式に記述したものである。例えば、この表50は図に示すように、T＝1から10までの範囲でTをインクリメントしながら、処理51と処理52を繰り返すといった内容になっている。処理51の内容は、ある演算処理ABCとする。また、処理52の内容は、パラメータX，Yの内容に応じて4種類に分けられている。このような表形式のプログラム表記は、例えば次のような基本形を組み合わせて行う。図7に示す処理の内容は次の基本形の説明によって明らかにする。

【0020】図8には、基本形T1と基本形T2とを示した。図8（a）は、基本形T1の内容を示し、左側が条件記述欄、右側が処理内容の記述欄を示している。即ち、図の例では、Xが1のときと2のときで処理内容を分けている。Xが1のときは処理ABCを実行し、Xが2のときは処理DEFを実行する。

【0021】（b）は基本形T2の内容を示し、XとYをパラメータとしてそれぞれその組合せによって処理の内容を切り換えている。即ち、Xが1でYが1の場合には、処理ABCを実行し、Xが1でYが2の場合には、処理DEFを実行する。また、Xが2でYが1の場合には、処理GHiを実行し、Xが2でYが2のとき、処理JKLを実行する。（a）に示す例の場合、ソースコードで記述しても表現上大きな差はないが、（b）のような内容になると、ソースコードで記述した場合とこのような表形式で記述した場合とでは、まず一見して理解し易いかどうかに大きな差がある。しかも、表形式の場合、全ての条件を網羅し易く、記入漏れや記述ミスが生じにくい。

【0022】図9には、基本形T3，T4，T5の説明図を示す。（a）に示す基本形T3では、表を用いて繰り返し演算を表現している。即ち、T＝1から10までの範囲でTをインクリメントし、処理ABCと処理DEFとを実行するといった内容になっている。特に、このような表の中に更に表を埋め込むネスト表現を実行する場合、ソースプログラムコードでは、繰返し命令、繰返し演算処理の仕切り、その他が入り乱れて、解釈も容易でなく煩雑になる。しかしながら、このような表形式で表現する場合には、図のABCという処理の中にこの図と同様の繰返し表現をはめ込めばよいだけで、どの繰返し表現がどの繰返し表現中に含まれるかあるいはどの繰返し表現が何回繰返されるか等、極めて容易にビジュアルに認識し区別できるといった効果がある。

【0023】（b）に示す基本形T4は、条件がアンドやオアで組み合わされているとき、その条件に合えばいくつかの処理を合わせて実行するといった表現を例示している。即ち、ここでは、Xが“1”でYが“1”の両方の条件が成立した場合に、処理ABCと処理DEFとを順に実行する。なお、図（c）に示す基本形T5は、これまでそれぞれ組合せとして説明した1個の欄に1個の処理の内容を記述したもので、無条件に処理ABCを実行するといった表現となっている。ほとんど全てのプログラムは、上記のような表を組み合わせ連結することで表現できる。しかも、これにより分かり易く論理的なミスを生じにくいといった特徴を持つ。

【0024】〈マトリクスコンパイラの機能〉マトリクスコンパイラは、マトリクスエディタ2が生成したプログラム辞書から図1に示すようなC言語によるソースプログラムコード11、メカ制御専用言語によるソースプログラムコード12、シミュレーション実行形式コード13を生成する部分である。

【0025】このマトリクスコンパイラ4は、一種のプロダクションシステムである。各生成対象毎に、各コード生成のための規則をプロダクションルールの形で持つ。即ち、図1に示すコード生成規則群6には、C言語によるソースプログラムコード11を生成するためのC言語ソースプログラムコード生成規則と、メカ制御専用言語によるソースプログラムコード12を生成するためのメカ制御専用言語プログラムコード生成規則8と、シ

ミュレーション実行形式コード１３を生成するためのシミュレーション実行形式コード生成規則９とを参照する構成になっている。各生成規則は、ＩＦ部とＴＨＥＮ部とから構成されている。ＩＦ部にはマッチング条件のパターンリストを含める。ＴＨＥＮ部には当該条件にマッチした場合のソースプログラムコード等への変換方法を記述する。その具体的内容は後で説明する。

【００２６】即ち、マトリクスコンパイラ４は、例えばマトリクスエディタ２により生成されたプログラム辞書をＣ言語によるソースプログラムコード１１に変換する場合には、Ｃ言語ソースプログラムコード生成規則７を参照しながら変換処理を進める。即ち、プログラム辞書の各項目と同一のパターンとなるＩＦ部を含む生成規則を、Ｃ言語ソースプログラムコード生成規則７から検索する。そして、該当する規則が見つかった場合、そのＴＨＥＮ部のソースプログラムコードを出力として得る。こうして、プログラム辞書をＣ言語によるソースプログラムコード１１に変換できる。メカ制御専用言語によるソースプログラムコード１２の生成やシミュレーション実行形式コード１３の生成も同様にして行うことができる。

【００２７】なお、以下の例では、ファームウェア全般にプログラム言語として汎用され、その中で最も普及しているＣ言語を例にして説明を行う。しかしながら、このような言語の如何を問わず、マトリクスコンパイラは対応するコード生成規則があれば、任意のソースプログラムコードを得ることが可能である。

【００２８】〈シミュレーションデバッガの機能〉シミュレーションデバッガ１４はマトリクスコンパイラ４が生成したシミュレーション実行形式コード１３を解析し実行するインタプリタである。このようなシミュレーションデバッガ１４は、従来よりプログラム開発支援に不可欠なものとして広く使用されてきている。その具体的な方法としては、プログラムを１ステップずつ実行させ結果を出力させたり、あるいはプログラム中にブレイクポイントを指定しそこまでプログラムを実行させその時点での出力や状態を表示させるといった用いられる。

本発明では、このようなシミュレーションデバッガを更に進め、メカ制御用ソフトウェアのデバッグにより実用的なメカ動作のアニメーション表示を表示できる構成にしている。図１は、このシミュレーションデバッガの構造を示した。このシミュレーションデバッガは、状態表示部６７を備えている。動作規則６１やシミュレータ動作条件６３等を元にメカ制御状態６６を生成する部分でメカ制御プログラム状態部６９に表示する部分である。なお、このような表示内容は、表示内容定義６８等を参照して決定する。

【００３０】即ち、状態更新部６４は、シミュレーション実行形式コードを１命令ずつ実行すると、その都度その時点でのメカ制御プログラム状態６５、メカ状態６６にメカ動作規則６１を適用しながら、その内容を更新していく。これによって、１ステップ毎にこのような状態を監視できる。また、ブレーク条件６３を判定して、ブレークポイントにおいて、オペレータからの入力を待つ。状態表示部６７は、例えばメカ状態６６の内容を参照し、メカの動きや状態等を表示内容定義６８に従ってアニメーション等を用いて表示する。即ち、例えばカード等の媒体を搬送路上で搬送するようなプログラムの場合、プログラムが１ステップ実行される度に、その際のカードの位置を搬送路も含めてそのままビジュアルに表示する。これによって、プログラマーはメカ状態が一見して把握でき、プログラム開発が容易になる。なお、メカ動作規則６１は、例えばステッピングモータの動作条件、ステッピングモータ同士の連結関係、センサ情報等を定義した内容のものである。

【００３１】（具体的な動作例）図１１に、具体的な操作画面例（その１）を示す。この図に示す操作画面は、先に説明したマトリクスエディタ２によりモジュール定義図を入力する場合の画面を示す。図に示すように、モジュール定義図を入力する場合には、このような操作画面を利用して、入力３１、モジュール３２、出力３３等の定義を記入する。なお、これはソースプログラムコード生成のみならずプログラム作成者本人あるいは第三者がプログラムの内容を容易に理解できるようにするための記述として利用される。

【００３２】図１２には、具体的な操作画面例（その２）を示す。この画面には、プログラムの実行手順を示す具体的な表が図示されている。ここでは、伝票とか通帳のような媒体を吸入してこれが搬送路上を搬送されている状態を検出し、必要な情報を印字して排出するといったメカニズム制御のためのプログラムを示している。このモジュールは、媒体の吸入処理に関する操作を示す部分である。その処理は表の上から順に実行される。即ち、最初のステップＳ１においては、媒体の状態を媒体無しというように初期化しておく。

【００３３】そして、次のステップＳ２は媒体が無い状態で媒体吸入を検出するための処理を行う。これは、一定の時間おきに媒体無しの状態が続く間繰り返し実行される。まず、ステップＳ３で、センサ状態をセンスする。このセンサは媒体搬送路上に設けられた特定のセンサである。そして、その結果を利用して次のステップＳ４の処理を行う。ステップＳ４の処理はセンサ１とセンサ２の状態に応じて４通りに分かれている。両方のセンサがＯＦＦの場合は何もしない。センサ１がＯＦＦでセンサ２がＯＮの場合には、媒体の状態をクランプという

状態に切り換える。ＭＧ２と指定したマグネットをＯＮする。即ち、マグネットをＯＮすることによって媒体をクランプして媒体の搬送を実行する。一方、センサ１がＯＮでセンサ２がＯＦＦの場合には、ＭＧ２と表示したマグネットをＯＮし、次にそのマグネットをＯＦＦする。また、センサ１も２もＯＮの場合には、同様にＭＧ２と表示したマグネットをＯＮし、その次にこれをＯＦＦする。

【００３４】このように、搬送路上のマグネットのＯＮとＯＦＦを繰り返し、媒体をつかんで搬送路上に吸入する。次に、ステップＳ５において、搬送起動を行う。ここでは、媒体の先端をどこまで送り込むか、先端目標位置や媒体の後端目標位置、最大移動量等を規定した上で、搬送機構を起動させる。次のステップＳ６では、モータの状態が動作中の間、ステップＳ７に示すような手順が実行される。即ち、ここでは媒体先端位置をステッピングモータに供給されるパルス数等でカウントし、その数が１２２７に達すると、記号ＭＧ３と表示したマグネットをＯＮする。その他の場合は、ＭＧ３と表示したマグネットをＯＦＦにしておく。

【００３５】以上のように、表形式によれば、非常に簡単に正確なプログラミングが可能となる。図１３には、シミュレーションデバッガの出力例説明図を示す。上記のような媒体の搬送等のメカニズムを制御するプログラムでは、その動作を解析する場合、媒体の位置、搬送路の状態、センサ等の状態をプログラムの任意のステップで一見して理解できるように表示することが好ましい。そこで、本発明のシミュレーションデバッガは、図１３に示すような搬送路７０とその搬送路のスケール７１とを表示する。１番上に示したのは搬送路、２番目に示したのは搬送路のスケールで、搬送路のスケール７１はステッピングモータに供給されるパルス数を基準にして表示されている。

【００３６】また、搬送路７０上には、媒体７２の位置を検出するセンサＭＳ２，ＭＳ４，ＭＳ７等が設けられている。更に、媒体７２を搬送駆動するローラを制御するためのマグネットＭＧ２，ＭＧ３，ＭＧ４等が設けられている。これらの各マグネットがＯＮＯＦＦする状態も、この図の下側に示すようにタイムチャートで表示される。タイムチャートのロウレベルはＯＦＦ、ハイレベルはＯＮの状態である。シミュレーションデバッガが各処理ステップ毎に、そのステップが終了したときの状態を図に示すような要領でアニメーション化する。そして、プログラマーの希望に従って表示する。これにより、媒体の挿入、検知、搬送、印字制御等を直接図解して見ることができ、プログラム製作作業の効率化が図られる。

【００３７】図１４には、図１２のＳ１に示した部分からソースプログラムコード出力を得るための具体的なマトリクスコンパイラの動作説明図を図示した。図の（ａ）はプログラム辞書、（ｂ）は日本語変換辞書、（ｃ）はマトリクスコンパイラのルール（シミュレーション実行形式コード生成規則）、（ｄ）はソースプログラムコード出力を示している。図１２のＳ１に示した命令は、「媒体状態＝媒体無し」という命令である。これは、即ち「媒体状態」という変数に対し媒体無しというデータを入力するための命令である。このような表形式のデータは、図１４（ａ）に示すように記述される。図の１行目のｎａｍｅ　＄ｗｈａｔは、この表の名称を表している。また、次の行のｔｙｐｅ　ｉｆ－ｗｈａｔは、表の種類を示している。次のｔ－ｗｉｄｔｈ　１６８　ｔｏｐは、この表の幅を示している。

【００３８】そして、この表の欄に記入された文字は５行目の“”に挟まれた媒体状態＝媒体無しという文字で表現されている。表の内容を記述する方法はどのような方法でもよいが、このような記述によって表のデータがコンピュータの認識できる文字データとなる。従って、逆にこのプログラム辞書を読み取ることによって、図１２に示すような表が描かれる。このようなプログラム辞書の作成方法自体は従来より表計算ソフトウェア等でよく知られており、どのような方法を使用しても差し支えない。

【００３９】ここで、このプログラム辞書からソースプログラムコードを得るために、（ｃ）に示すマトリクスコンパイラのルールが適用される。ところが、ソースプログラムコードに日本語を含めてよい場合と、日本語は使用できない場合とがある。いずれの場合にもソースプログラムコードの生成を可能にするためには、プログラム辞書に含まれている日本語を英語等に変換しておくことが好ましい。このために、（ｂ）に示す日本語変換辞書が使用される。従って、プログラム辞書の中で「媒体状態」という言葉と「媒体無し」という言葉が英語に変換される。

【００４０】日本語変換辞書はこのような目的のために、表の中で使用される言葉に対応する英語を用意する。図の矢印によって示した「媒体状態」という単語は、ｂ＿ｓｔａｔｕｓという英語に置き換えられる。「媒体無し」という単語はこの日本語変換辞書には用意されていない。そこで、マトリクスコンパイラは自動的にこの媒体無しという言葉に対応する変数を他の言葉と重複しないように設定してしまう。例えば、その変数をＪｇ６７と定める。このような準備を行った後、（ｃ）に示すルールが適用される。マトリクスコンパイラには多数のルールが設けられており、先に説明したように、そのＩＦ部が適用される場合に、ＴＨＥＮ部に置き換えられたソースプログラムコードが出力される。

【００４１】この（ｃ）の１行目は表の名前で、（ａ）に示した２行目と一致する。［ＩＦ部には％ｋｅｙという言葉がある。これは１つの命令であって、その次に続くｂｏｄｙという言葉を見つけなさいという命令である。

同様にして、ｔｈｅｎという言葉とｗｈａｔという言葉を続けて見つけなさいという命令が続く。そして、ｗｈａｔという言葉を見つけた場合に、これに続く言葉を＆ｒｅｓｔという変数に代入しなさいという命令になる。これによって＆ｒｅｓｔという変数に「媒体状態＝媒体無し」という部分が代入される。なお、これは、先に説明したように、先にｂ＿ｓｔａｔｕｓ＝Ｊｇ６７という式に変換されている。

【００４２】ここで、（ｃ）に示すＴＨＥＮ部が実行される。即ち、ここでは、ａｐｐｅｎｄという命令によって＆ｒｅｓｔという変数の中身に“；”を後ろに付けて出力するといった内容になる。これは、即ち、（ｄ）に示すような式をソースプログラムコードとして出力することになる。これによって、図１４（ａ）に示すようなプログラム辞書からこの（ｄ）に示すようなソースプログラムコードが得られる。他の部分についても全く同様にプログラム辞書に含まれた日本語を一旦英語に変換し、その後マトリクスコンパイラに用意された各種のルールに従って各部を順次変換することによってソースプログラムコードが得られる。

【００４３】図１５は日本語変換辞書の一部の例を示し、図１６～図２０は図１２の表に対応するその他の部分のプログラム辞書の例を示す。また、図２１，図２２は、図１２のＳ４の部分を変換するために使用されるルールのリストであり、図２３，図２４は、マトリクスコンパイラの出力として得られたソースプログラムコードの例を示す。プログラム辞書中、Ｓ４と示した部分は、図２１，図２２に示すマトリクスコンパイラのルールに従って変換され、図２３，図２４に示すＳ４と印をしたソースプログラムコードに置き換えられる。その手順等は先に説明した通りである。

【００４４】本発明は以上の実施例に限定されない。上記実施例においては、メカ動作制御のためのプログラム作成に本発明を利用した例を示したが、この他、例えばワードプロセッサのような文書データの加工処理、その他各種の情報処理のためのプログラム作成に利用できる。また、プログラム言語はＣ言語を用いて説明したが、Ｃ言語の改良型であるＣ＋＋、その他Ｃ言語系のプログラムやこれ以外の各種の汎用されているプログラム作成に利用することが可能である。

【００４５】

【発明の効果】以上説明した本発明のプログラム開発支援システムによれば、プログラムのロジックの表記を表形式により分かり易く行い、これを自動的にプログラムソースコードに変換する構成にしたので、プログラムの表現について専門的な知識を持たないものにも分かり易い誤りの無いプログラムの記述ができる。また、第三者も容易に分かり易いプログラムの記述が可能となる。更に、プログラムの各命令が実行されたときに、制御対象装置の各部の状態を表す状態情報等を具体的に表示するシミュレーションデバッガを用意することによって、プログラムの開発をより容易に具体的に行うことが可能になる。

【図面の簡単な説明】

【図１】本発明のプログラム開発支援システム実施例を示すブロック図である。

【図２】本発明の実施のためのハードウェア説明図である。

【図３】モジュール定義図の説明図である。

【図４】データ形式定義図の説明図である。

【図５】構造体型定義図の説明図である。

【図６】データ意味定義図の説明図である。

【図７】表の例説明である。

【図８】表の基本形説明図（その１）である。

【図９】表の基本形説明図（その２）である。

【図１０】シミュレーションデバッガの構造を示すブロック図である。

【図１１】具体的な操作画面例（その１）である。

【図１２】具体的な操作画面例（その２）である。

【図１３】シミュレーションデバッガの出力例説明図である。

【図１４】具体的なマトリクスコンパイラの動作説明図である。

【図１５】日本語変換辞書の例説明図である。

【図１６】プログラム辞書の例説明図（その１）である。

【図１７】プログラム辞書の例説明図（その２）である。

【図１８】プログラム辞書の例説明図（その３）である。

【図１９】プログラム辞書の例説明図（その４）である。

【図２０】プログラム辞書の例説明図（その５）である。

【図２１】マトリクスコンパイラのルールの例説明図（その１）である。

【図２２】マトリクスコンパイラのルールの例説明図（その２）である。

【図２３】ソースプログラムコード出力の例説明図（その１）である。

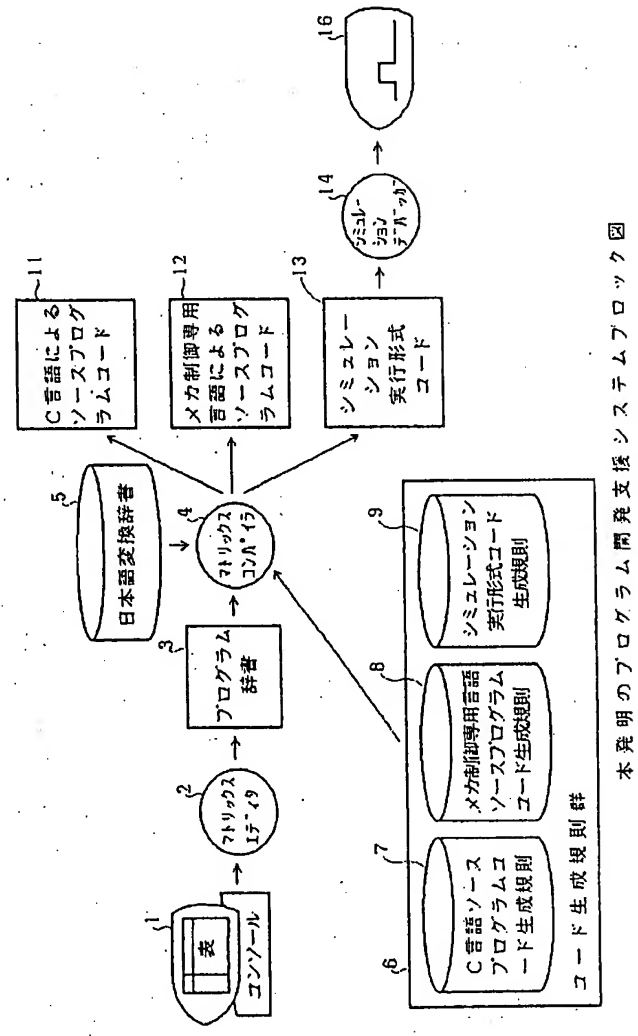
【図２４】ソースプログラムコード出力の例説明図（その２）である。

【符号の説明】

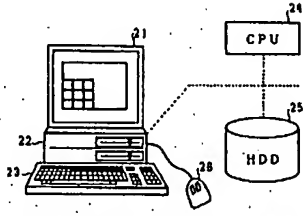
１　コンソール
２　マトリクスエディタ
３　プログラム辞書
４　マトリクスコンパイラ
６　コード生成規則群
１１　Ｃ言語によるソースプログラムコード
１２　メカ制御専用言語によるソースプログラムコード

１３　シミュレーション実行形式コード　　１４　シミュレーションデバッガ

【図１】

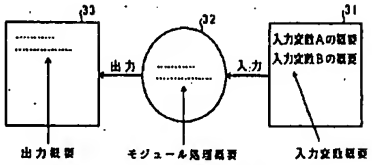


【図2】



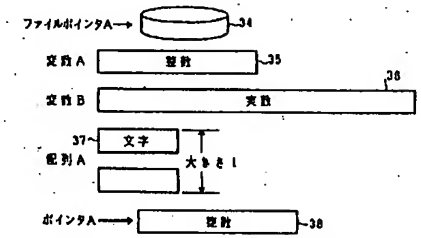
本発明の実施のためのハードウェア説明図

【図3】



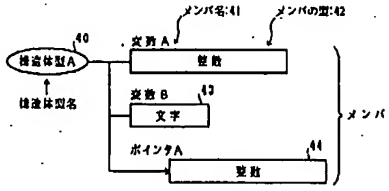
モジュール定義図の説明図

【図4】



データ形式定義図の説明図

【図5】



構造体型定義図の説明図

【図6】

出力値の概要

センサの出力	
値	内容
1	ON
0	OFF

データ意味定義図の説明図

【図7】

T＝1からT<＝10の間T++しながら		
ABC		
X＼Y	1	2
1	ABC	DEF
2	GHI	JKL

表の例説明図

【図8】

X	処理
1	ABC
2	DEF

X＝1のとき、処理ＡＢＣを実行
X＝2のとき、処理ＤＥＦを実行

基本形Ｔ1
（a）

X＼Y	1	2
1	ABC	DEF
2	GHI	JKL

X＝1，Y＝1のとき、処理ＡＢＣを実行
X＝1，Y＝2のとき、処理ＤＥＦを実行
X＝2，Y＝1のとき、処理ＧＨＩを実行
X＝2，Y＝2のとき、処理ＪＫＬを実行

基本形Ｔ2
（b）

表の基本形説明図（その１）

【図9】

T＝1からT<＝10の間T++しながら
ABC
DEF

T＝1からT<＝10の間、
処理ＡＢＣと処理ＤＥＦを実行し、
Tを1インクリメントする

基本形Ｔ3
（a）

条件	処理
(X==1) かつ (Y==1)	ABC
	DEF

X＝1かつY＝2のとき、
処理ＡＢＣと処理ＤＥＦを実行

基本形Ｔ4
（b）

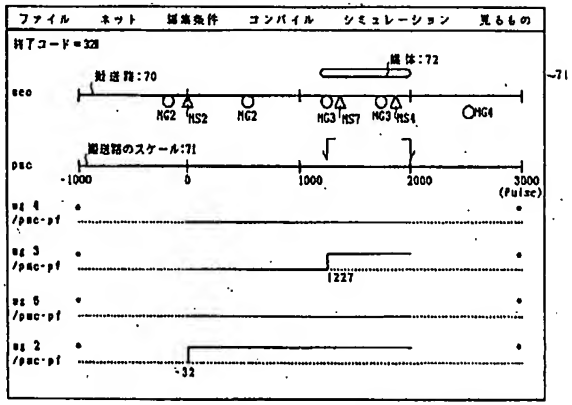
ABC

無条件に処理ＡＢＣを実行

基本形Ｔ5
（c）

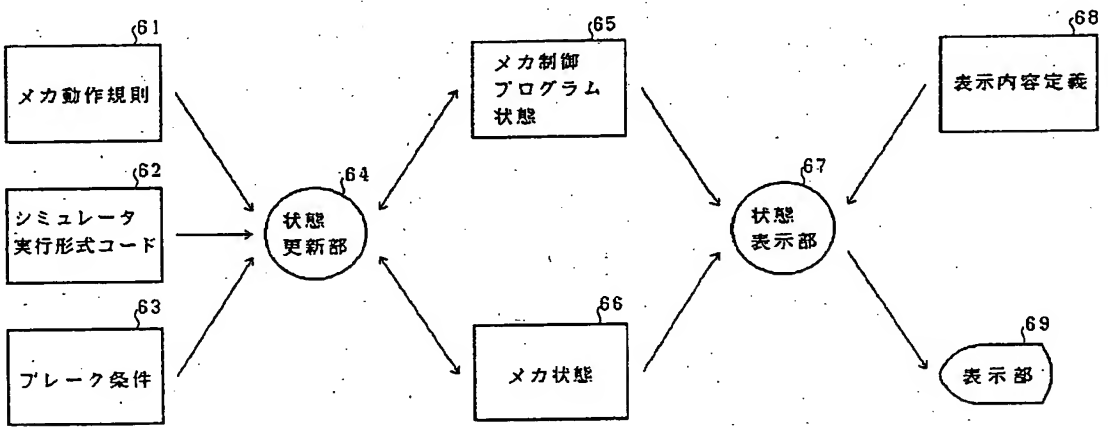
表の基本形説明図（その２）

【図13】



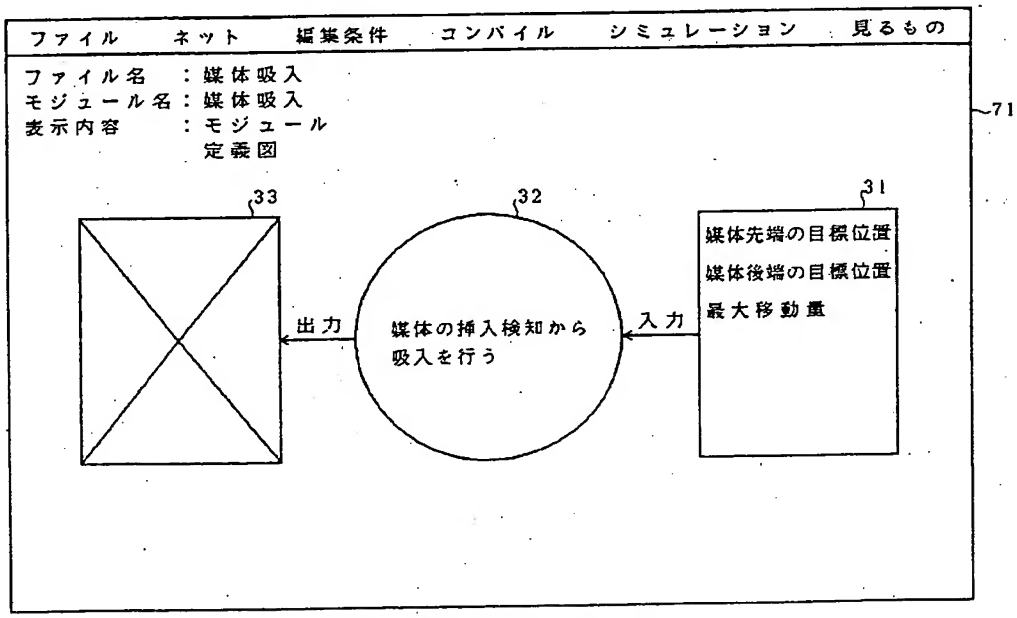
シミュレーションデバッガ出力例

【図10】



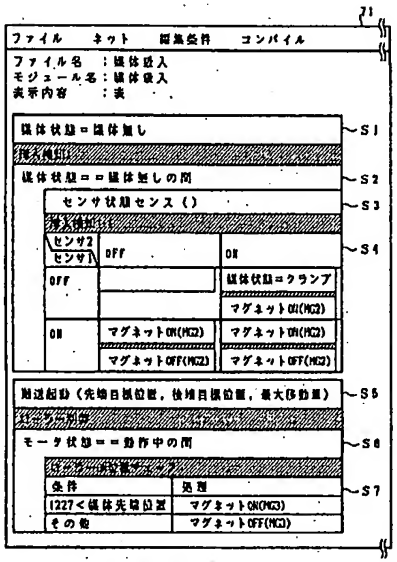
シミュレーションデバッガ構造

【図11】



具体的な操作画面例（その１）

【図12】



具体的な操作画面例（その２）

【図17】

```
((prog-name "媒体搬送")
 (define ((name "マクロ定義")
          (type macro)
          (t-width 144 top)
          (body (var (width 80 1)
                     (what "マクロ名"))
                (proc))
          (body (value (width 48 top)
                       (what "ON"))
                (then (width 40 top)
                      (what "1")))
          (body (value (width 56 top)
                       (what "OFF"))
                (then (width 40 top)
                      (what "0")))
          (body (value (width 96 top)
                       (what "媒体無し"))
                (then (width 40 top)
                      (what "0")))
          (body (value (width 96 top)
                       (what "クランプ"))
                (then (width 40 top)
                      (what "1")))
          (body (value (width 56 top)
                       (what "MG2"))
                (then (width 40 top)
                      (what "2")))
          (body (value (width 56 top)
                       (what "MG3"))
                (then (width 40 top)
                      (what "3")))))
 (var ((name "モータ状態")
       (type "short")
       (sel "外部変数")
       (pointer nil)
       (ini nil))
      ((name "媒体先端位置")
       (type "整数")
       (sel "自動変数")
       ⋮
```

プログラム辞書例（その２）

【図14】

(a)

```
(ptable ((name $what)
         (type if-what)
         (t-width 168 top)
 (body (then (width 168 1)
       (what "媒体状態＝媒体無し"))))
```

プログラム辞書

⋮

(b)

```
(停止                 "stop" )
(１パルス吸入         "kyunyu_1pls" )
(１パルス排出         "haisyutu_1pls" )
(媒体状態             "b_status" )
(媒体先端位置         "pf" )
(媒体後端位置         "pt" )
```

⋮

日本語変換辞書

(c)

```
(if-what
(IF
((%key body(%key then(%key what&rest?then)&etc)&etc)))
(THEN
(%eval(append(quote?then)'(";")))))
```

マトリクスコンパイラのルール

(d)

```
b_status=jg67;
```

ソースプログラムコード出力

【図15】

```
(整数              "int")
(文字              "char")
(実数              "float")
(符合なし整数      "unsigned int")
(符合なし文字      "unsigned char")
(符合なし short    "unsigned short")
(倍長実数          "double")
(メイン            "main")
(構造体            "struct")
(リターン          "return")
(ブレーク          "break")
(自動変数          nil)
(外部変数          "extern")
(静的変数          "static")
(レジスタ変数      "register")
(その他            "default")
(改行              "printf(¥"¥¥n¥")")
(表示              "printf")
(オープンできません。  "Can't open file!")
(事象              "EVENT")
(状態              "STATUS")
(オン              "1")
(オフ              "0")
(先端位置          "front_pos")
(センサ１          "sensor_1")
(センサ２          "sensor_2")
(センサー          "sensor")
(マトリックス      "matrix")
(クランプ          "crump")
(排出              "haisyutu")
(吸入終了位置      "kyunyu_end_pos")
(停止              "stop")
(１パルス吸入      "kyunyu_1pls")
(１パルス排出      "haisyutu_1pls")
(媒体状態          "b_status")          → これが説明に対応する項目です。
(媒体先端位置      "pf")
(媒体後端位置      "pt")
(先端位置リード    "read_f_pos")
(動作中            "01")
(モータ状態        "motor_status")
(センサーセンス    "sensor_sence")
(ビット０          "0x0001")
```

日本語変換辞書の例説明図

【図16】

〈〈〈データ形式〉〉〉

```
全体   = ((prog_name "プログラム名１")          ... プログラム名
        (define ("マクロ定義" (macro -----))) ... マクロ定義
        (var -------)                         ... グローバル変数定義
        (struct ----)                         ... 構造体定義
        (module ----))                        ... モジュール定義

module = ((name "モジュール名１")             ... モジュール名
        (type "...")                          ... モジュールの型
        (abstruct "...")                      ... モジュールの概要
        (input --------)                      ... 入力変数定義
        (output -------)                      ... 出力値定義
        (var ----------)                      ... ローカル変数定義
        (struct -------)                      ... 構造体定義
        (ptable -------))                     ... 表の定義

struct = ((name "構造体名１")                 ... 構造体名
        (member
           ((name "メンバ名１")               ... メンバ名
            (type "...")                      ... メンバの型
            (sel ------)                      ...
            (pointer -)                       ... ポインタか？
            (length "...")                    ... 配列時の大きさ
            (struct-name "...")               ... 構造体型名
            (iml -------))                    ... メンバの値の意味
           ((name "メンバ名２")
            ..................)
            ..................)
         ..........)

var    = ((name "変数１")                     ... 変数名
          (type "...")                        ... 変数の型
          (sel "...")                         ... 記憶クラス
          (pointer -)                         ... ポインタか？
          (length "...")                      ... 配列時の大きさ
          (struct-name "...")                 ... 構造体型名
          (iml -----))                        ... 変数の値の意味

ptable = ((name "表の名前１")                 ... 表の名前
          (type "...")                        ... 表の形式
          (t-width 表の幅 何列目)             ... 表示情報
          (body --------))                    ... 表形式にあった情報
```

プログラム辞書例（その１）

【図18】

```
(output nil)
(var nil)
(struct nil)
(ptable ((name $what)
         (type if-what)
         (t-width 168 top)
         (body (then (width 168 1)
                     (what "媒体状態=媒体無し"))))
        ((name "押入検知1")
         (type if-while)
         (t-width 480 top)
         (body (var (until (width 176 2)
                           (what "媒体状態==媒体無し")))
               (then (ptable ((name $what)
                              (type if-what)
                              (t-width 176 4)
                              (body (then
                                     (width 176 1)
                                     (what "センサ状態センス()"))))
                             ((name "押入検知1-1")
                              (type if-vh)
                              (t-width 440 4)
                              (body (var
                                     (width 88 1)
                                     (what "センサ1")))
                              (body (value
                                     (width 56 1)
                                     (what "OFF"))
                                    (thenx
                                     (body
                                      (v-h
                                       (width 88 1)
                                       (what "センサ2")))
                                     (body
                                      (value-h
                                       (width 56 2)
                                       (what "OFF"))
                                      (then
                                       (ptable
                                        ((name $what)
                                         (type if-what)
                                         (t-width 24 2)
                                         (body (then
                                                (width 24 1)
                                                (what ""))))))))))))))
```

S1

S4

プログラム辞書例（その３）

【図19】

```
                (body
                 (value-h
                  (width 48 3)
                  (what "ON"))
                 (then
                  (ptable
                   ((name $what)
                    (type if-what)
                    (t-width 168 3)
                    (body (then
                    (width 168 1)
                    (what "媒体状態=クランプ"))))
                   ((name $what)
                    (type if-what)
                    (t-width 168 3)
                    (body (then
                    (width 168 1)
                    (what "マグネットON(MG2)")))))))))
(body (value
       (width 48 1)
       (what "ON"))
      (thenx
       (body
        (v-h
         (width 88 1)
         (what "センサ2")))
       (body
        (value-h
         (width 56 2)
         (what "OFF"))
        (then
         (ptable
          ((name $what)
           (type if-what)
           (t-width 168 2)
           (body (then
           (width 168 1)
           (what "マグネットON(MG2)"))))
          ((name $what)
           (type if-what)
```

S4

プログラム辞書例（その４）

【図２０】

```
    (t-width 176 2)
    (body (then
    (width 176 1)
    (what "マグネットOFF(MG2)")))))))
(body
 (value-h
  (width 48 3)
  (what "ON"))
 (then
  (ptable
   ((name $what)
    (type if-what)
    (t-width 168 3)
    (body (then
    (width 168 1)
    (what "マグネットON(MG2)"))))
   ((name $what)
    (type if-what)
    (t-width 176 3)
    (body (then
    (width 176 1)
    (what "マグネットOFF(MG2)")))))))))))))
```

S４

プログラム辞書例（その５）

【図２１】

```
(if-vh
 (IF
    ((%key body (%key var (%key what &rest ?var) &etc) &etc)
     (%pattern vh-values ?pattern1)))
 (THEN
    (%eval (append '("switch ")
                   (list (st-append2 (append '(" (") (quote ?var) '(")")))
                   '(" (")
                   (quote ?pattern1)
                   '((")")))))))

(vh-values
 (IF
  ((%pattern vh-value1 ?pattern1)
   (%pattern-nil vh-values ?pattern2)))
 (THEN
   (%eval (cons (quote ?pattern1) (quote ?pattern2)))))

(vh-value1
 (IF
  ((%key body (%key value (%key what &rest ?value) &etc)
              (%key thenx
                    (%key body (%key v-h (%key what &rest ?v-h) &etc) &etc)
                    (%pattern vh-value-h ?pattern1) &etc)
              &etc)))
 (THEN
   (%eval (append '("case ")
                  (quote ?value)
                  '(":")
                  (list (append '("switch ")
                                (list (st-append2
                                       (append '(" (") (quote ?v-h) '(")")))
                                '(" (")
                                (quote ?pattern1)
                                '((")")))
                        '("break;")))))))
```

S４を変換する為のルール（その１）

【図22】

```
(vh-value-h
 (IF
  ((%pattern value-h ?pattern1)
   (%pattern-nil vh-value-h ?pattern2) ))
 (THEN
   (%eval (cons (quote ?pattern1) (quote ?pattern2)))))

(value-h
 (IF
  ((%key body (%key value-h (%key what &rest ?value-h) &etc)
              (%key then (%key ptable ?ptable) &etc)
              &etc))))
 (THEN
   (%eval (append '("case ")
                  (quote ?value-h)
                  '(":")
                  (quote ?ptable)
                  '(("break;")))))))
```

S4を変換する為のルール（その２）

【図23】

```
#include <stdio.h>
#define ON 1
#define OFF 0
#define Jg67 0
#define crump 1
#define MC2 2
#define MC3 3
extern short motor_status;
unsigned short b_status, sensor_1, sensor_2;
int pf, pt;

void Jg70(Jg72, Jg77, Jg81)
int Jg72, Jg77, Jg81;
{
        b_status = Jg67;          S1
        while (b_status==Jg67){
                Jg85();
                switch (sensor_1){
                        case OFF:
                                switch (sensor_2){
                                        case OFF:
                                                ;
                                                break;
                                        case ON:
                                                b_status=crump;
                                                Jg86(MC2);
                                                break;
                                }
                                break;
                        case ON:
                                switch(sensor_2){
                                        case OFF:
                                          |
                                          |
                                          |
                                          |
```

ソースプログラムコード例（その１）

【図24】

```
b_status = Jg67;
while (b_status==Jg87){
        Jg85();
        switch (sensor_1){
                case OFF:
                        switch (sensor_2){
                                case OFF:
                                        ;
                                        break;
                                case ON:
                                        b_status=crump;
                                        Jg86(MG2);
                                        break;
                        }
                        break;
                case ON:
                        switch(sensor_2){
                                case OFF:
                                        Jg86(MG2);
                                        Jg87(MG2);
                                        break;
                                case ON:
                                        Jg86(MG2);
                                        Jg87(MG2);
                                        break;
                        }
                        break;
        }
}
Jg88(Jg72, Jg77, Jg81);
while(motor_status==01)
        if (1227<pf)
                Jg86(MG3);
        else
                Jg87(MG3);
}
```

S4

ソースプログラムコード例（その２）